Contents

Using an External HTTPS Proxy

5 minute read ✔ page test

See also

The Configure an Egress Gateway example shows how to direct traffic to external services from your mesh via an Istio edge component called *Egress Gateway*. However, some cases require an external, legacy (non-Istio) HTTPS proxy to access external services. For example, your company may already have such a proxy in place and all the applications within the organization may be required to direct their traffic through it.

This example shows how to enable access to an external HTTPS proxy. Since applications use the HTTP CONNECT method to establish connections with HTTPS proxies, configuring traffic to an external HTTPS proxy is different

from configuring traffic to external HTTP and HTTPS services.

Before you begin

- Setup Istio by following the instructions in the Installation guide.

The egress gateway and access logging will be enabled if you install the `demo` configuration profile.

- Deploy the `sleep` sample app to use as a test source for sending requests. If you have `automatic sidecar injection` enabled, run the following command to deploy the sample app:

```
$ kubectl apply -f @samples/sleep/sleep.yaml@
```

Otherwise, manually inject the sidecar before deploying the `sleep` application with the following command:

```
$ kubectl apply -f <(istioctl kube-inject -f @samples/sleep/sleep.yaml
@)
```

You can use any pod with `curl` installed as a test source.

- Set the `SOURCE_POD` environment variable to the name of your source pod:

```
$ export SOURCE_POD=$(kubectl get pod -l app=sleep -o jsonpath={.items
..metadata.name})
```

- Enable Envoy's access logging

Deploy an HTTPS proxy

To simulate a legacy proxy and only for this example, you deploy an HTTPS proxy inside your cluster. Also, to simulate a

more realistic proxy that is running outside of your cluster, you will address the proxy's pod by its IP address and not by the domain name of a Kubernetes service. This example uses Squid but you can use any HTTPS proxy that supports HTTP CONNECT.

1. Create a namespace for the HTTPS proxy, without labeling it for sidecar injection. Without the label, sidecar injection is disabled in the new namespace so Istio will not control the traffic there. You need this behavior to simulate the proxy being outside of the cluster.

    ```
    $ kubectl create namespace external
    ```

2. Create a configuration file for the Squid proxy.

```
$ cat <<EOF > ./proxy.conf
http_port 3128

acl SSL_ports port 443
acl CONNECT method CONNECT

http_access deny CONNECT !SSL_ports
http_access allow localhost manager
http_access deny manager
http_access allow all

coredump_dir /var/spool/squid
EOF
```

3. Create a Kubernetes ConfigMap to hold the configuration of the proxy:

```
$ kubectl create configmap proxy-configmap -n external --from-file=squ
id.conf=./proxy.conf
```

4. Deploy a container with Squid:

```
$ kubectl apply -f - <<EOF
apiVersion: apps/v1
kind: Deployment
metadata:
  name: squid
  namespace: external
spec:
  replicas: 1
  selector:
    matchLabels:
      app: squid
  template:
    metadata:
      labels:
        app: squid
    spec:
      volumes:
      - name: proxy-config
        configMap:
```

```
          name: proxy-configmap
      containers:
      - name: squid
        image: sameersbn/squid:3.5.27
        imagePullPolicy: IfNotPresent
        volumeMounts:
        - name: proxy-config
          mountPath: /etc/squid
          readOnly: true
EOF
```

5. Deploy the `sleep` sample in the `external` namespace to test traffic to the proxy without Istio traffic control.

```
$ kubectl apply -n external -f @samples/sleep/sleep.yaml@
```

6. Obtain the IP address of the proxy pod and define the `PROXY_IP` environment variable to store it:

```
$ export PROXY_IP="$(kubectl get pod -n external -l app=squid -o jsonp
ath={.items..podIP})"
```

7. Define the `PROXY_PORT` environment variable to store the port of your proxy. In this case, Squid uses port 3128.

```
$ export PROXY_PORT=3128
```

8. Send a request from the `sleep` pod in the `external` namespace to an external service via the proxy:

```
$ kubectl exec "$(kubectl get pod -n external -l app=sleep -o jsonpath
={.items..metadata.name})" -n external -- sh -c "HTTPS_PROXY=$PROXY_IP
:$PROXY_PORT curl https://en.wikipedia.org/wiki/Main_Page" | grep -o "
<title>.*</title>"
<title>Wikipedia, the free encyclopedia</title>
```

9. Check the access log of the proxy for your request:

```
$ kubectl exec "$(kubectl get pod -n external -l app=squid -o jsonpath
={.items..metadata.name})" -n external -- tail /var/log/squid/access.l
og
1544160065.248    228 172.30.109.89 TCP_TUNNEL/200 87633 CONNECT en.wi
kipedia.org:443 - HIER_DIRECT/91.198.174.192 -
```

So far, you completed the following tasks without Istio:

- You deployed the HTTPS proxy.
- You used `curl` to access the `wikipedia.org` external service through the proxy.

Next, you must configure the traffic from the Istio-enabled pods to use the HTTPS proxy.

Configure traffic to external HTTPS proxy

1. Define a TCP (not HTTP!) Service Entry for the HTTPS proxy. Although applications use the HTTP CONNECT method to establish connections with HTTPS proxies, you must configure the proxy for TCP traffic, instead of HTTP. Once the connection is established, the proxy simply acts as a TCP tunnel.

```
$ kubectl apply -f - <<EOF
apiVersion: networking.istio.io/v1beta1
kind: ServiceEntry
metadata:
  name: proxy
spec:
  hosts:
  - my-company-proxy.com # ignored
  addresses:
  - $PROXY_IP/32
  ports:
  - number: $PROXY_PORT
    name: tcp
    protocol: TCP
  location: MESH_EXTERNAL
EOF
```

2. Send a request from the `sleep` pod in the `default` namespace. Because the `sleep` pod has a sidecar, Istio controls its traffic.

```
$ kubectl exec "$SOURCE_POD" -c sleep -- sh -c "HTTPS_PROXY=$PROXY_IP:
$PROXY_PORT curl https://en.wikipedia.org/wiki/Main_Page" | grep -o "<
title>.*</title>"
<title>Wikipedia, the free encyclopedia</title>
```

3. Check the Istio sidecar proxy's logs for your request:

```
$ kubectl logs "$SOURCE_POD" -c istio-proxy
[2018-12-07T10:38:02.841Z] "- - -" 0 - 702 87599 92 - "-" "-" "-" "-"
"172.30.109.95:3128" outbound|3128||my-company-proxy.com 172.30.230.52
:44478 172.30.109.95:3128 172.30.230.52:44476 -
```

4. Check the access log of the proxy for your request:

```
$ kubectl exec "$(kubectl get pod -n external -l app=squid -o jsonpath
={.items..metadata.name})" -n external -- tail /var/log/squid/access.l
og
1544160065.248    228 172.30.109.89 TCP_TUNNEL/200 87633 CONNECT en.wi
kipedia.org:443 - HIER_DIRECT/91.198.174.192 -
```

Understanding what happened

In this example, you took the following steps:

1. Deployed an HTTPS proxy to simulate an external proxy.
2. Created a TCP service entry to enable Istio-controlled traffic to the external proxy.

Note that you must not create service entries for the external services you access through the external proxy, like `wikipedia.org`. This is because from Istio's point of view the requests are sent to the external proxy only; Istio is not aware of the fact that the external proxy forwards the requests further.

Cleanup

1. Shutdown the `sleep` service:

```
$ kubectl delete -f @samples/sleep/sleep.yaml@
```

2. Shutdown the `sleep` service in the `external` namespace:

```
$ kubectl delete -f @samples/sleep/sleep.yaml@ -n external
```

3. Shutdown the Squid proxy, remove the `ConfigMap` and the configuration file:

```
$ kubectl delete -n external deployment squid
$ kubectl delete -n external configmap proxy-configmap
$ rm ./proxy.conf
```

4. Delete the `external` namespace:

```
$ kubectl delete namespace external
```

5. Delete the Service Entry:

```
$ kubectl delete serviceentry proxy
```